Progresivní technologie v informatice I

Umělá inteligence a strojové učení 1

Fakulta informačních technologií

České vysoké učení technické v Praze

2023/2024

Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Hlavní body

Příklad: prenatální (1/4)

- S prvním *datovým modelem* jste se pravděpodobně setkali ještě v prenatálním období Vašeho života.
- Při odhadování porodní váhy plodu pomocí ultrazvuku (tzv. SONO) se používá mj. následující model:

Model pro porodní váhu (Shephard et al.)

$$\log_{10}(eFW) = -1.749 + 0.017 \cdot BPD + 0.005 \cdot AC - \frac{2.646}{1000} \cdot (BPD \cdot AC),$$

eFW = odhadovaná hmotnost při narození, BPD = biparietal diameter (příčný průměr hlavy), AC = abdominal circumference (obvod břicha).

- Pro nás to bude **lineární regresní model** (7. přednáška), kde je
  - porodní váha eFW tzv. **vysvětlovaná proměnná** (angl. **target variable**),
  - BPD, AC a $BPD \cdot AC$ jsou pak tři tzv. **příznaky** (anlg. a často i česky **features**),
  - čísla $1.749, 0.017$ atd. jsou takzvané **regresní koeficienty** (příp. váhy), obecně **parametry modelu**.

Příklad: prenatální (2/4)

Model pro porodní váhu (Shephard et al.)

$$\log_{10}(eFW) = -1.749 + 0.017 \cdot BPD + 0.005 \cdot AC - \frac{2.646}{1000} \cdot (BPD \cdot AC),$$

- **Jak se tento model používá?**
  1. Na ultrazvuku se změříte veličiny BPD a AC,
  2. získaná čísla dosadíte do pravé strany vzorce,
  3. výsledek je odhad hodnoty $\log_{10}(eFW)$ a tedy i kýžené porodní váhy eFW.

- **Kde se vzala ta divná čísla jako 2.646 apod.?**
  To se právě budeme učit v tomto kurzu: Jsou to parametry zvoleného modelu a ty se vždy nějakým způsobem odhadují na základě dostupných dat, to je tzv. **učení modelu**.

Příklad: prenatální (3/4)

Jak asi pan Shephard et al. došli právě k tomuto modelu:

1. Z nějakého důvodu věřili, že příznaky *BPD* a *AC* jsou pro porodní váhu důležité. Nejspíše ale zkoušeli i jiné, ale ty buď nepřinášely velké zlepšení nebo se daly *BPD* a *AC* plně nahradit.

2. Pomocí různých testovacích procedur si jako model zvolili lineární regresní model s třemi příznaky a čtyřmi koeficienty:

$$\log_{10}(eFW) = w_0 + w_1 \cdot BPD + w_2 \cdot AC + w_3 \cdot (BPD \cdot AC).$$

3. Předchozí fázi, kdy hledáme „tvar“ modelu, se říká **ladění hyperparametrů** (angl. **hyperparameter tuning**).

4. Koeficienty w_i pak odhadli z dat o již narozených dětech, u kterých měli přesnou porodní váhu i prenatálně naměřené hodnoty *BPD* a *AC*.

Příklad: prenatální (4/4)

Zmíněná data, na kterých se model učil (a testoval), mohla vypadat nějak takto:

`kid_id`	*FW*	*BCD*	*AC*
1	číslo	číslo	číslo
2	číslo	číslo	číslo
⋮	⋮	⋮	⋮

Pro zajímavost (detaily v 7. přednášce):

- Označme sloupec pod *FW* jako vektor $\vec{Y}$
- a vytvořme matici $\mathbf{X}$ o čtyřech sloupcích, kde v každém řádku je vektor $(1, BCD, AC, BCD \cdot AC)$.
- Nejpoužívanější metoda pro výpočet koeficientů w_i je **metoda nejmenších čtverců**, ta nám říká, že

$$(w_0, w_1, w_2, w_3)^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{Y}.$$

- Tento vzorec jsme získali vyřešením jistého optimalizačního problému (a.k.a. *hledání extrémů*), což je velice častý případ: **učení modelu = optimalizace**!

Supervizované vs. nesupervizované učení

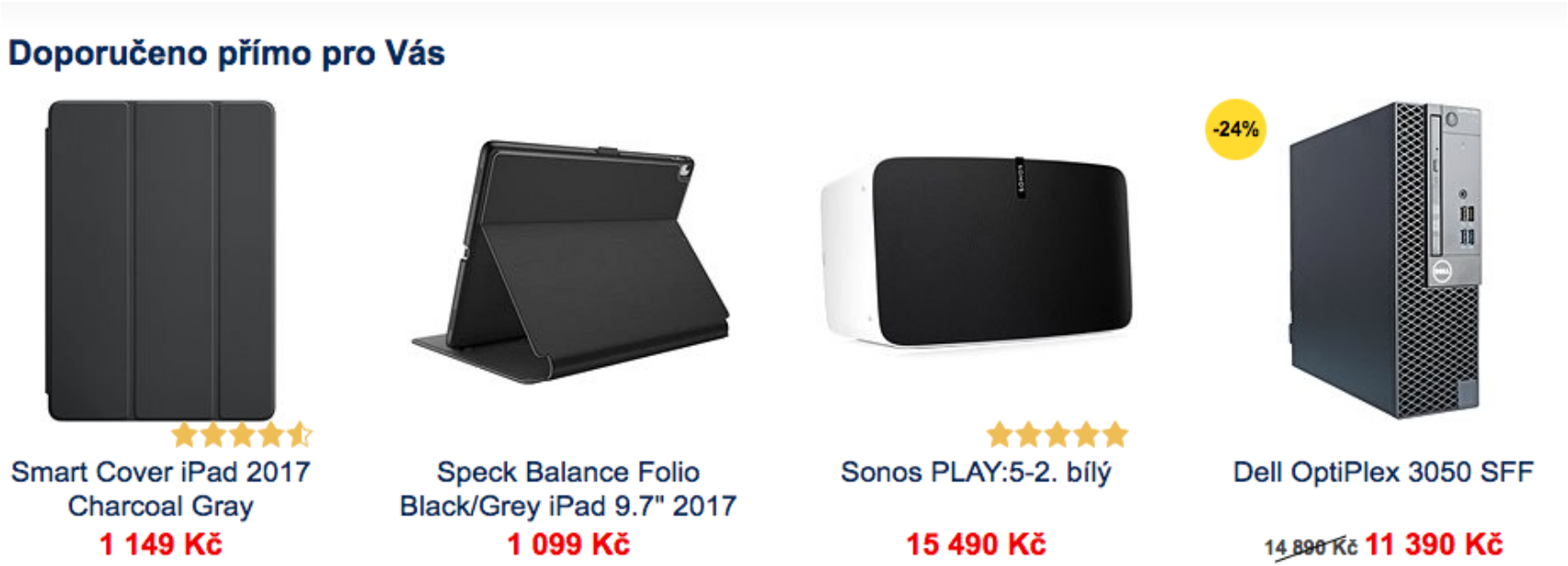
- Předchozí *prenatální* příklad je typickou ukázkou **supervizovaného učení** (učení s učitelem, angl. **supervized learning**).
- Tím „učitelem“ jsou zde známé hodnoty porodních vah u dětí, což je veličina, kterou se snažíme pomocí modelu *predikovat* resp. pochopit, na čem závisí.
- Někdy takovou veličinu ale ani nemáme a prostě se v datech pokoušíme nějak vyznat a najít jejich skrytou strukturu.
- Takovým problémům se říká **nesupervizované učení** (učení bez učitele) a typickým příkladem je **clusterování** dat (téma 4. přednášky).

Příklady nesupervizovaného učení

- Problém clusterování je velice obvyklý v praxi.
- Pokud máte například e-shop (nebo banku, nebo telefonního operátora), chcete se vyznat ve svých zákaznících, o kterých máte nasbíraná různá data (tzv. *customer segmentation*).
- Můžete tak hledat např. podmnožinu „nejlepších“ zákazníků, kterým má cenu věnovat speciální péči. Nebo naopak skupinu, která potřebuje k polepšení pomoci nějakou reklamní akcí (cílení reklamy je velký byznys).
- Do nesupervizovaného učení také (obvykle) spadá i **detekce anomálií** (angl. **anomaly detection**).
- Např. banka se snaží najít podezřelé transakce (fraud detection, ochrana proti zneužití karty, atp.).

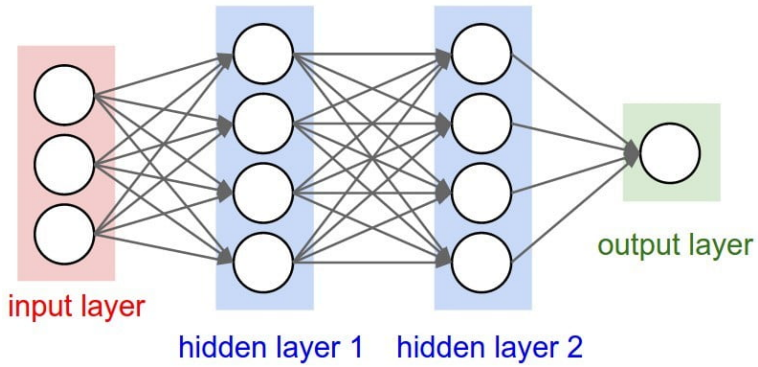
Další příklady: doporučovací systémy

- Dalším příkladem problému řešeného pomocí zkoumání dat je tzv. **doporučování** (angl. **recommendation**).
- Například: vlastníte-li e-shop (příp. internetový časopis, iTunes, Netflix atp.), snažíte se na základě dat o zákaznících a zejména zákazníkovi, který právě prohlíží Vaše stránky, odhadnout, co by si tak mohl ještě chtít koupit (přečíst, podívat, poslechnout) a to mu ukázat.



Další příklady: data bez jasných příznaků

- Data ale vždy nemusí mít formu tabulky. Může se jednat o obrázky, videa, časové řady, dlouhé texty atp., ze kterých je těžké získat pro modely příznaky.
- V takovém případě si musíte dát práci a nějaké příznaky z dat vydolovat (tzv. **feature extraction**).
- Nebo použijete algoritmy a metody, které si příznaky vytvářejí samy automaticky.
- Mezi takové metody patří (čím dál populárnější) umělé **neuronové sítě** (angl. artificial **neural networks**, ANN)
- O ANN budeme mluvit v posledních přednáškách. Používají se k všemožným úkolům (překlady, detekce objektů v obrázku videu, hraní GO, clusterování, detekci anomálií, . . .).



Hlavní body

Co je problém klasifikace

- Supervizované učení: Snažíme se zjistit, jak vysvětlovanou proměnnou Y ovlivňují příznaky $X_0, X_1, \ldots, X_{p-1}$, hledáme tedy nějaký funkční vztah tak, aby „co nejvíce platilo“

$$Y \approx f(X_0, X_1, \ldots, X_{p-1}).$$

- Funkce f nemusí být nutně podobná funkcím, které znáte z analýzy. Např. v této přednášce to bude strom.
- Tvar hledané funkce často ovlivňuje to, jakých hodnot může nabývat vysvětlovaná proměnná Y:
  - Může-li nabývat jen několik málo hodnot, mluvíme o problému **klasifikace** (angl. **classification**). Sem spadá např. určení, jestli pacient má/nemá nemoc, jaké písmeno je (ručně) napsáno na obrázku, atp.
  - Může-li nabývat tolika hodnot, že je rozumnější ji považovat za *spojitou*, mluvíme o problému **regrese** (angl. **regression**).
- **Rozhodovací stromy** (angl. **decision trees**) lze použít pro oba typy problému: my začneme tím klasifikačním.

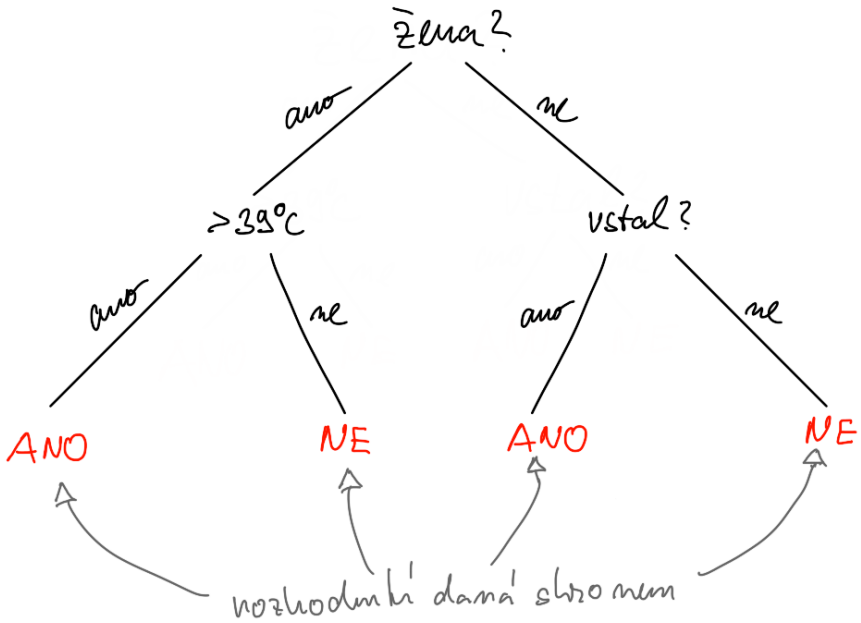
Ukázka použití rozhodovacího stromu (1/6)

- Velmi často je klasifikační problém **binární**, kdy proměnná Y může mít jen dvě hodnoty.
- My si použití stromu ukážeme na (vymyšlených) datech a problému určování, jestli pacient má či nemá závažnou nemoc známou jako „rýmička“.
- Příznaky budou pro jednoduchost také binární: Pohlaví (žena/muž), horečka ($> 39°\mathrm{C}/\leq 39°\mathrm{C}$) a to, jestli daný člověk zvládl/nezvládl vstát z postele.
- Ukážeme si dva rozhodovací stromy a porovnáme si, jak je který z nich dobrým modelem následujících dat:

rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?
ano	muž	ne	ne
ne	žena	ano	ano
ne	muž	ne	ano
ano	žena	ano	ne

Ukázka použití rozhodovacího stromu (2/6)

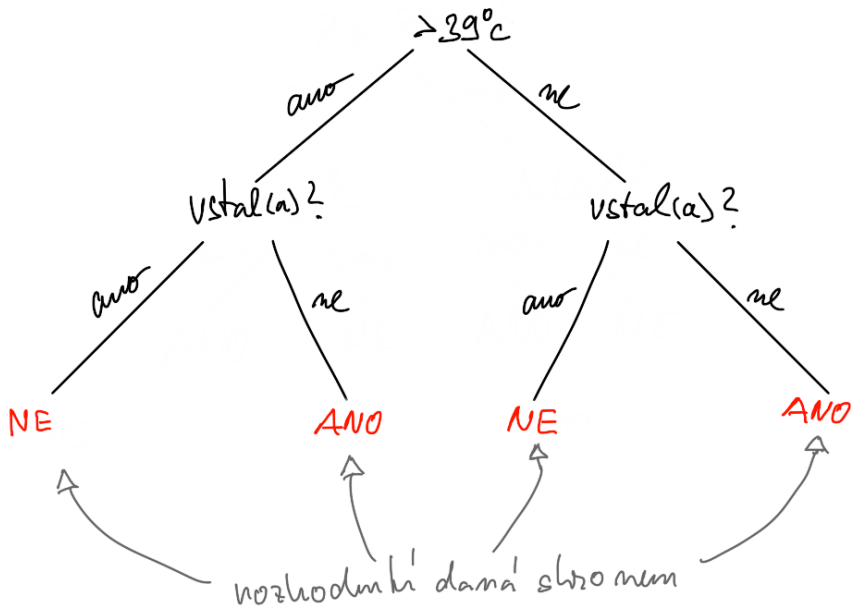
Strom 1:



rýmička	pohlaví	$> 39°C$	vstal(a)?	co říká strom
ano	muž	ne	ne	ano
ne	žena	ano	ano	ano
ne	muž	ne	ano	ne
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (3/6)

Strom 2:



rýmička	pohlaví	$> 39°C$	vstal(a)?	co říká strom
ano	muž	ne	ne	ano
ne	žena	ano	ano	ne
ne	muž	ne	ano	ne
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (4/6)

- Strom 1 dává špatný výsledek pro druhý řádek, strom 2 dává správné výsledky pro všechny řádky.
- Strom 2 je tedy, zdá se, lepší. Je však toto dostatečné zdůvodnění?
- My ve skutečnosti chceme vědět, jak často se strom trefí **pro všechna možná data**.
- Což je trochu smůla, protože všechna možná data nikdy nemáme. Máme většinou jen „jednu tabulku“ dat a ta nám musí stačit jak pro vytvoření (neboli naučení) stromu, tak i pro ověření toho, jak je dobrý.
- Jak se to dělá, si ukážeme později.

Ukázka použití rozhodovacího stromu (5/6)

- Strom 1 i strom 2 jsou stromy hloubky 2 a mají tedy 4 listy.
- Kdybychom tedy vytvořili strom hloubky 3, měl by 8 listů. Přesně tolik je ale taky možných kombinací hodnot tří příznaků! **To už není model, ale index!**
- Strom hloubky tři by se tedy mýlil pouze v případě, že by hodnoty všech příznaků byly stejné, ale hodnota vysvětlované proměnné by byla jiná (např. kdyby jeden pacient byl chlapík s angínou):

rýmička	pohlaví	$> 39°C$	vstal(a)?
⋮	⋮	⋮	⋮
ano	muž	ano	ne
ne	muž	ano	ne
⋮	⋮	⋮	⋮

- V takovém případě by se mýlil libovolný strom a dokonce i jakákoli funkce příznaků (viz definici funkce).

Ukázka použití rozhodovacího stromu (6/6)

- Skutečným model rýmičky je, jak známo, toto: rýmička nastává právě když

$$(\text{žena} \ \wedge (> 39^\circ) \wedge \ \text{nevstala}) \vee (\text{muž} \ \wedge ((> 39^\circ) \vee \ \text{nevstal}))$$

- Takový model ale není postižitelný stromem hloubky dva! Vzpomeňme BI-MLO a minimalizaci formulí v disjunktivním normálním tvaru ...

Hlavní body

1. ML a AI : Základní přehled
2. Klasifikace – ukázka na rozhodovacím stromě
3. Trénovací a testovací data: konstrukce a hodnocení modelu

Jak je můj strom dobrý?

Nyní se dostáváme k druhé otázce: **Jak poznám, že můj vytvořený strom není jen dobrým modelem dat, která mám, ale bude dobře fungovat i pro data jiná?**

- Chceme-li rozhodovat, jak je model dobrý, potřebujeme nějakou objektivní vyčíslitelnou míru jeho kvality.
- Volba této míry je důležitou součástí celého procesu hledání modelu. Míry se obvykle velmi liší pro problémy klasifikace a regrese.
- My se nyní věnujeme klasifikaci a vystačíme si s přirozenou mírou **klasifikační přesnosti** (angl. **classification accuracy**), která prostě měří poměr správně klasifikovaných, tedy je rovna číslu (příp. procentu)

$$\frac{\text{počet správně klasifikovaných dat}}{\text{počet všech dat}}.$$

- Např. rozhodovací strom zkonstruovaný algoritmem ID3 pro množinu osmi dat se pletl ve dvou případech, a tedy jeho přesnost byla $\frac{6}{8} = 0.75 = 75\ \%$.

Jak je můj strom dobrý? Jakože opravdu?

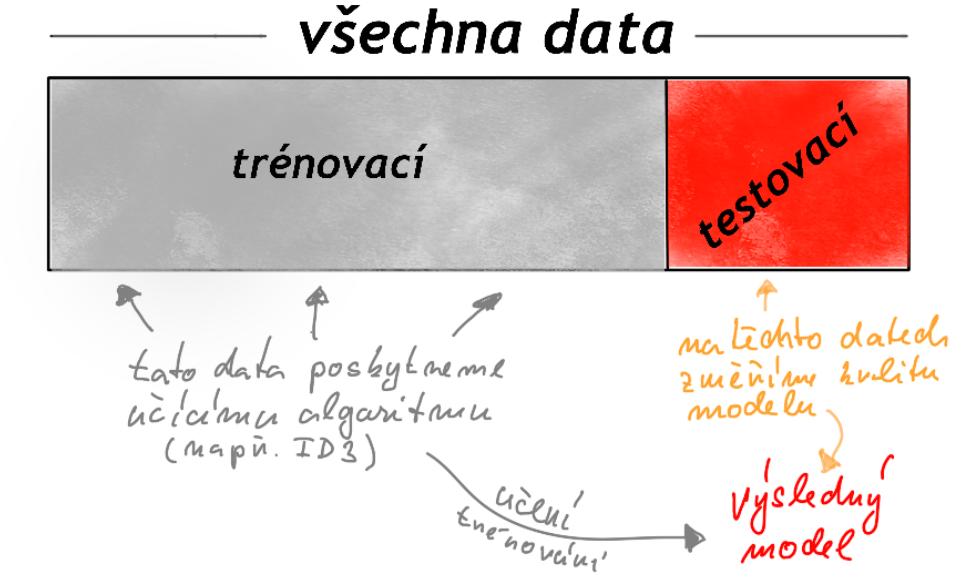
- Mohlo by se zdát, že máme vyřešeno: prostě zkonstruujeme strom, který má pro naše data nejvyšší přesnost.
- S tímto přístupem bychom ale narazili, neb by vedl k hlubokým stromům, které mají na listech třeba jen jeden datový bod.
- Platí totiž, že čím je strom hlubší, tím má lepší přesnost!
- My vlastně nechceme maximalizovat přesnost na našich datech, ale **na všech možných datech**, která ovšem nemáme.

(?) Jak to vyřešit?

- Uděláme to tak, že naše data rozdělíme na dva kusy: na jednom (tom větším), model naučíme (např. C4.5 algoritmem) a na tom druhém kusu změříme přesnost. **Tak dostaneme spolehlivější odhad toho, jak se bude náš model chovat pro data, na kterých se neučil.**

Trénovací a testovací data

- Těm dvěma kusům dat se obvykle říká **trénovací** a **testovací data** (angl. **train** a **test set**).
- Chybovost modelu (pro nás nepřesnost = 1 - přesnost) na těchto množinách dat se pak adekvátně říká **trénovací** a **testovací chyba** (angl. **train** a **test error**).



Přeučení modelu (1/2)

- V případě stromů zjevně platí, že čím hlubší strom učíme, tím dostaneme menší trénovací chybu. Spolehlivější měrou kvality je však testovací chyba, neb tou trénovací si lžeme do kapsy: Měříme, jak dobře jsme *přizpůsobili* strom dostupným datům, nikoli jak dobře jsme odhadli skrytý model za těmito daty schovaným (pokud tedy na takový model věříte).

- Tomuto přílišnému přizpůsobení trénovacím datům se říká **přeučení** modelu (angl. **overfitting**).

- Obvykle platí, že čím složitější model (v našem příp. čím hlubší strom), tím nižší trénovací chyba.

- Testovací chyba se však chová jinak: Nejdříve se zvětšováním složitosti modelu klesá, ale v jistý okamžik začne růst. Najít tento bod zlomu je úkolem celého procesu budování modelu.

Přeučení modelu (2/2)

- Následující obrázek ukazuje typický vývoj trénovací a testovací chyby v závislosti na hloubce stromu (`max_depth`).
- Z tohoto obrázku je vidět, že nejrozumnější volba parametru `max_depth` je 6.